第47卷 第10期
2015年10月

哈尔滨工业大学学报
JOURNAL OF HARBIN INSTITUTE OF TECHNOLOGY

Vol. 47 No. 10
Oct. 2015

doi: 10.11918/j.issn.0367-6234.2015.10.019

星载大容量固态存储器快速可靠启动算法设计

李 姗[1,2], 宋 琪[1,2], 朱 岩[1], 安军社[1]

(1.中国科学院 空间科学与应用研究中心, 100190 北京; 2. 中国科学院大学, 100049 北京)

摘 要: 为解决星载 SSR 启动缓慢,可靠性不足所带来的无法满足复杂、灵活的任务需求问题,研究星载大容量固态存储器(SSR)的索引建立机制,分析传统方案的优缺点并结合在轨运控需求、工作模式等特点,提出一种适合基于 NAND FLASH 的星载大容量固态存储器的快速启动算法.原有文件系统存储索引表来自于 NAND FLASH 空余区,算法增加新的保留区设计及相应的启动过程,避免重启时扫描空余区重新建立索引.针对空间环境单粒子效应带来的存储错误,NAND FLASH 保留区的索引信息采取 ECC 编码、冗余备份、分区存储等可靠性措施,提高索引表的可靠性.本文介绍了应用保留区的启动工作机理以及不同模式下的更新方式,阐述了系统在功能性重启和故障性重启下的扫描方式,并建立数学模型分析算法的有效性,最后在使用欧比特 NAND FLASH 搭建的测试平台上进行验证.算法功能性重启索引建立耗时 5.643 ms,故障重启索引建立 73.985 ms,而传统算法重启索引建立 50.37 s.实验结果表明,本算法显著减少了系统启动耗时.

关键词: 快速启动; NAND FLASH; 固态存储器; 存储系统

中图分类号: TP274 **文献标志码:** A **文章编号:** 0367-6234(2015)10-0100-06

Design of quick initialization algorithm for space-borne solid state recorder

LI Shan[1,2], SONG Qi[1,2], ZHU Yan[1], AN Junshe[1]

(1. Center for Space Science and Application Research, Chinese Academy of Sciences, 100190 Beijing, China;
2. University of Chinese Academy of Sciences, 100049 Beijing, China)

Abstract: The starting up of onboard SSR is slow and the management is unreliable ,they can not satisfy the demand of the mission which is becoming more and more complex and flexible. To solve the problem, this paper focuses on the index establishing mechanism for onboard Solid State Recorder (SSR), a quick initialization method for onboard massive capacity SSR based on NAND FLASH is proposed by analyzing the advantages and disadvantages of traditional scheme, the requirement of on-orbit operation and control, and working mode, etc. The proposed method uses the design of reserved area and corresponding booting process on the basis of the original NAND FLASH indexing mechanism by space area, which largely reduced the organization time for index table. By considering the memory error caused by single particle effect, the index information in the reserved area of NAND FLASH uses reliability methods including ECC coding, redundancies and partition storage to improve the reliability of the index table. The restart working mechanism and the update of different mode for the reserved area of application is firstly introduced. Followed by the description of the scanning mode in both functional restart and fault restart, and a mathematical model is built to prove the effectiveness, and is verified in a platform established using Orbita NAND FLASH. Finally, in the designed platform the functional restart of the algorithm takes 3.643 ms and the fault restart takes 73.985 ms while the traditional algorithm takes 50.37 s, which demonstrate that the proposed algorithm significantly reduces the mounting up time.

Keywords: quick initialization; NAND FLASH; solid state recorder; storage system

星载大容量固态存储器(SSR)是航天器的关键设备之一,由于地面接收站数量、接收范围有限,航天器在轨运行时,产生的大量珍贵数据需先在星上暂存,待入境后进行数据传输.航天器工作环境恶劣,空间中充斥着大量带电粒子可导致设备内管理逻辑混乱,部分单元失效甚至引起设备起火烧毁整个航天器[1-5].异常发生时,需通过地面在过境时发送指令以备份单元切换的方式降低或消除带电粒子带来的效应的影响.另外,对绕月、绕火星等非地球

收稿日期: 2014-05-23.
基金项目: 中国科学院战略性先导科技专项资助(XDA04060300).
作者简介: 李 姗(1989—),女,博士研究生;
朱 岩(1973—),男,研究员,博士生导师;
安军社(1969—),男,教授,博士生导师.
通信作者: 李 姗,lishanmg@gmail.com.

恒星的飞行器而言，其测控和数传接收还受到各星球间位置关系变化的影响，因此星载 SSR 管理算法对可靠性和启动时间都提出了较高的要求.

NAND FLASH 的存储密度高，低功耗，抗震能力强等[6-8]. 目前，常见的 NAND FLASH 管理方法有 JFFS[9]，YAFFS[10]等. 其中 YAFFS 是针对 NAND FLASH 设计的管理方法，启动较 JFFS 更快[11]，应用范围更广. YAFFS 算法中索引表使用多级映射方式，系统重建耗时长，风险大. Yim[12]提出的快照技术(Snapshot) 将系统内数据索引记录下来并保存在 NAND FLASH 中，系统再次启动时只需读取快照内数据即可重建索引，但该算法的快照机制只在系统正常关机时才启动，异常关机时快照内容无法重建系统，必须对所有 NAND FLASH 空余区进行扫描，依旧存在启动时间过长的问题. 星载 SSR 的启动时间由索引表建立时间、CPU 读取索引表时间以及存储系统初始状态设置时间 3 部分构成. 256 Gb 容量(32 K 条索引表项) 情况下，CPU 读取索引表时间在 1 s 以内，初始状态设置过程在几个 CPU 周期之内即可完成，而索引表的建立时间占启动时间的 95%以上. 因此，本文重点讨论如何对索引表建立时间的优化设计.

1　NAND FLASH

NAND FLASH 是一种层次结构的掉电非易失存储器，页作为基本单元构成块，多个块组成芯片. 每页包含数据区和空余区，空余区用于存储一些管理信息. NAND FLASH 由 FN tunnel 构成，它是芯片中的最小存储单元，可以通过放电的方式使其为零，但无法单独使其置 ‘1’，只能整块充电置 ‘1’. 因此，NAND FLASH 不能直接更新，需要先擦除后更新. NAND FLASH 的基本操作包括: 读取，擦除，写入. 其中读取和写入的基本单元为页，而擦除的基本单元为块[13]. NAND FLASH 采用数据地址分时复用的形式，通过 IO 分时传输数据、地址及控制信息. 坏块是一些无法被彻底擦除或者其中有些位无法翻转块的统称. 由于成本和工艺限制，出厂时 NAND FLASH 中会存在一些坏块，称为初始坏块. 为区分这些出厂坏块，厂家通常在 NAND FLASH 空余区的第一个字节标记非“0XFF”. NAND FLASH 的块擦除极限为5 000~100 000 次，因此会有坏块在使用过程中产生，称作使用坏块.

为满足空间应用需要，原始 NAND FLASH 芯片不仅经过抗辐照加固而且使用 3-D 封装技术进行叠装，以实现更高的存储密度[14]. 3-D 技术将 8 片 NAND FLASH 基片(DIE) 封装为成一个镀金立方体，将芯片的控制信号，电源线连接在一起，而数据线、片选信号分别引出. 以珠海欧比特公司的 VDNF64G08 为例[15]，每个芯片包含 8 个 die，每个 die 包含 4 096 个块，每块包含 64 页，每页包含 4 KBytes数据区和 128 Kbytes 空余区.

2　快速启动算法设计

星载 SSR 在轨运行时有 3 种工作模式: 存储模式、回放模式以及混合模式. 索引表随星载 SSR 内数据存储状态变化持续更新，索引表更新策略需灵活适应各工作模式的需求. 传统应用中采用页作为索引最小单元，不仅占用大量存储空间且启动缓慢[15]. 现在越来越多的研究者把眼光集中在以块为单元甚至更大的单元索引策略上[16]. NAND FLASH 读操作的最小单元是页，擦除操作的最小单元是块. 因此，本文采用以块为主以页为辅的索引方式，有效提高启动速率.

2.1　星载 SSR 的启动基本流程

传统星载 SSR 在对 NAND FLASH 进行数据存储、读取等操作的同时，将管理信息存储在 NAND FLASH 空余区. 由于 NAND FLASH 单页加载时间较长，约为 200 ~700 μs，而一页数据加载时间通常不超过 130 μs，为了提高 SSR 的吞吐量，多采用多级流水操作[17]. 空余区采用页为单元标记，当页操作完毕后根据操作结果在页空余区中写入页信息，包括页使用情况、页属性、流水级信息等. 空间中存在多种高能粒子，易诱发 NAND FLASH 发生单粒子效应，产生逻辑错误或引起功能异常，严重影响航天器在轨效能的发挥. 为提高索引条目的可靠性，加入 ECC 校验码，可实现自动纠正 1 位错误，检测 2 位错误. 具体页信息存储情况如表 1 所示.

表 1　NAND FLASH 空余区页记录信息表

页种类	空余区 第一字节	空余区 第二、三字节	空余区 第四字节	空余区 第五字节
已使用	页属性信息	文件号	流水级信息	ECC 校验码
未使用	全 ‘1’	全 ‘1’	全 ‘1’	全 ‘1’

星载 SSR 启动时遍历所有页的空余区，并将页信息整合为按照块为单位的新条目. 块条目包含块信息、使用信息等. 块条目结构组成如表 2 所示.

表 2　块条目信息表

块种类	第 0- 11 位	第 12- 15 位	第三、四 字节	第五 字节	第六 字节
已使用	块地址	块属性信息	文件号	起始地址	ECC 校验码
未使用	块地址	全 ‘1’	全 ‘1’	全 ‘1’	全 ‘1’

索引表建立完成之后，由 CPU 读取并存入缓存 SDRAM 中，供 CPU 对存储区进行管理. 因此，对于传统算法管理的星载 SSR 而言，索引表一份存储在 CPU 缓存 SDRAM 中，另一份存储在 NAND FLASH 的空余区. 第一份掉电丢失，启动时必须遍历所有块空余区，致使启动时间随容量增加大幅增长[12]，无

法满足星载 SSR 的应用需求.

2.2 保留区索引表设计

为提高系统启动速度,本文提出保留区概念,在星载 SSR 存储区中划分一块区域专门保存索引信息,定义为保留区索引表.启动时,优先搜索保留区索引表,集中搜索区域降低搜索用时.

保留区索引表基于 CPU 中原始索引表架构,精简组织信息,减少组织用时,减小索引表,提高启动速度.根据保留区索引条目类型将其划分为坏块信息区与数据信息区,由于二者更新频率、方式不同,因此将两者独立存储、管理.坏块区更新频率低,可靠性要求更高,对坏块信息区进行双份冗余存储.数据信息区保存索引条目,坏块区保存坏块块号.索引表条目可用数组表示为:{块号,块类型,文件号,文件占用页数,ECC 编码},索引条目使用固定的位数存储,替代条目标示符,提高检索效率.保留区中,块的数据区存储索引信息,空余区存储维护信息.空余区第一个字节代表块属性,第二个字节代表页状态,块属性分为正常块(0x “03”)、未使用块(0x “FF”),页状态分为有效(0x”55”)、无效(0x”00”).

随着星载 SSR 的运行,索引表中将产生大量失效条目,需定期整理.如将整个索引表读出,剔除失效条目,再写入,需占用 NAND FLASH 总线较长时间,降低系统存储效率.为此本文提出分 die 管理的方式,将保留区分 die 划分,每个 die 的保留区中仅存储该 die 的索引表,使用时只需维护当前 die,充分利用空间,降低各 die 间的耦合度,提高可靠性同时增强算法适用性.

首次上电,对所有 NAND FLASH 芯片进行扫描得到出厂坏块地址,并分别将其全部写入每个 die 保留区的第一个 block 坏块区中,然后使用 Page Cope-Back 将其中内容拷贝至第二个 block 的坏块区中,并在第一块空余区写入 0x “55”,标示该页为有效的索引表存储页.正常工作时,CPU 按一定更新周期对保留区索引表进行更新,将新增的数据信息写入数据信息区.根据 NAND FLASH 的特性,无法对写入块直接更新但分析可知再次写入,对应位为两次写入量相与的结果,因此可以通过再次写入零的方式将对应位写为零,此操作定义为覆写.新增信息如为文件写入,顺序添加条目; 如为文件擦除,将对应条目的文件大小覆写为 0; 如为擦除坏块则将其块号写入坏块信息存储页; 如为出错块则先将其写入数据信息存储页中,待失效处理后将地址写入坏块区.

每个 Die 的保留区采用固定定位的方式,提高系统故障后重启的可靠性.每个 Die 的前 5 个好块定为保留区,随着系统信息的更新保留区内也可能出现坏块,如出现坏块顺序向后扩展保留区,保证保留区中至少有 3 个好块可供使用.更新策略也采用顺序更新方式,即第 i 块是两个区都更新,第 $i+1$ 块是独立更新坏块区.

索引更新策略需配合各工作模式以保证性能最优.存储模式下数据流量低,NAND FLASH I/O 总线空闲时间充裕,保留区正常更新.回放模式和混合模式下数据流量高,NAND FLASH I/O 总线忙碌,停止更新保留区,仅实时更新原始索引表和备份索引表.在境内时,地面会根据需求,发送指令标记不再需要的文件为失效数据.索引条目对应文件失效时,将该条目对应文件占用页数改写为零,该操作定义为失效处理,失效处理通过覆写即可快速实现.出境模式后,需根据地面指令更新索引表.通常,过境后会有大量数据被标记无效.为提高更新效率,将保留区中入境前有效的 block 标记无效,根据原始索引表中失效信息对保留区组织新索引表写入保留区中空白块.采用流水线方式更新索引表,提高整体更新效率,具体流程如图 1 所示.其中 N_{c1} = 更新总级数; N_c = 单级更新总数; N_{e1} = 擦除总级数; N_e = 单级擦除总数; 一个 die 内有 B 个块.

3 星载 SSR 的快速启动模式

星载 SSR 在首次启动或者彻底复位后的首次启动需要遍历 Flash 空余区,无法实现快速启动.而在一般工作过程中重启时能够使用快速启动模式.实际应用中,快速启动拥有两种模式,一种为功能性重启,一种为故障性重启.功能性重启为恢复功能或消除某些粒子效应时进行的软件重启.故障性重启多为系统在运行中遇到的不可抗力使得其突然掉电或者复位.两者间最大的区别为功能性重启时过程可控,系统有足够的时间将缓存区的索引表写入保留区并将保留区索引表所在位置保存,而故障性重启时过程不可控.

功能性重启: 由计算机中主控计算机板发送重启指令,星载 SSR 系统中计算机板接到指令后优先将缓存中索引信息存入保留区.保存保留区索引表后,星载 SSR 系统中 CPU 将保留区索引表所在位置发送给主控计算机板然后重启.再次上电,CPU 只扫描保留区有效块的数据区即可重建索引.

故障性重启: 系统突然掉电丢失 CPU 缓存内信息,包括原始索引表,保留区索引表更新信息,最后更新位置以及整体分布位置.再次启动后,重建步骤如下:

1) 定位保留区. CPU 按顺序读取每个 die 内前两块空余区前两个字节,搜索到 0x”0155”即停止,如无则从该 die 末尾开始倒序搜索,搜索到 0x”0155”即止;

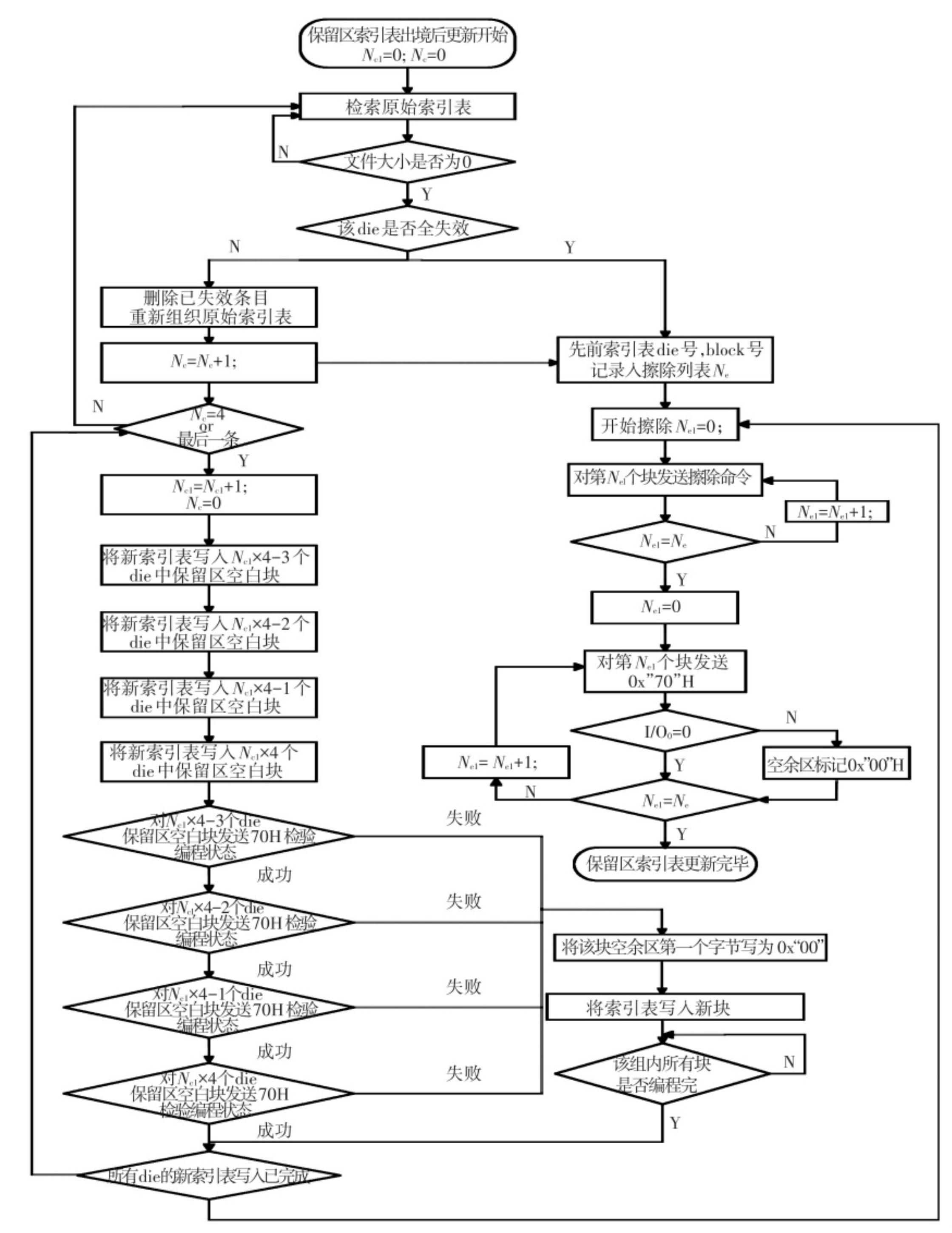


图 1　索引表更新流程

2）读取保留区. FPGA 顺序读取每个 die 保留区有效块的空余区，扫描0x”0155”. 根据 0x”0155”的位置定位有效页，读取有效页的数据区；

3）定位最后更新位置. 读取每个 die 索引表记录中最后一块的下一非坏块空余区，如非 0x”FF”，则表明此块已使用. 块内实际使用情况与索引表记录不符，此 die 为最后更新位置；

4）恢复更新信息. 假设最后更新 die 的索引表最后一条地址是第 m 块，更新周期为 N 块. 将从 m 块开始扫描每页空余区的前 6 字节，直至 $m+N$ 块或者扫描到某页空余区第一个字节为“FF”表明此块未写入过数据为止.

4　系统性能分析

4.1　基本假设

本算法的启动时间与 NAND FLASH 大小、保留区内坏块的数量、索引表的大小、更新方式、故障发生时缓存区内容大小、索引表更新周期都相关. 为计算方便，做出如下假设：

1）系统寿命内每个 die 内坏块的数量不超过总量的一半；

2) 目标 NAND FLASH 满足:
$1\text{page} = (D+S)\text{Byte}$, $1\text{block} = P_{\text{pages}}$, $1\text{die} = B_{\text{blocks}}$. 测试系统包括 Q_d 个 die;

3) 保留区内有效块为第 i 块的概率为 $n_i\%$;

4) 读取时钟周期为 t_{clk}, NAND FLASH 内部加载时间为 t_{busy}, 读取比特数为 q_{read}, 因此, 读 NAND FLASH 的时间可以表示为

$$T_{\text{read}} = 7t_{\text{clk}} + t_{\text{busy}} + q_{\text{read}}t_{\text{clk}}. \quad (1)$$

令 $T_c = 7t_{\text{clk}} + t_{\text{busy}}$. 式(1) 可简化为

$$T_{\text{read}} = T_c + q_{\text{read}}t_{\text{clk}}.$$

5) 最后更新位置为第 N_l 块, $0 \leqslant N_l \leqslant Q_d$;

6) 算法更新周期为 N_{up};

7) 第 i 个 die 索引表大小为 V_i, $0 \leqslant V_i \leqslant P(D+S)$.

4.2 快速启动、传统索引建立时间分析及对比

根据以上假设建立 NAND FLASH 数学模型, 重点对本文提出的快速启动算法的故障性启动时间 T_{startn} 和传统算法的启动时间 T_{starto} 进行计算.

T_{startn} 的计算根据启动流程可分为如下 4 步:

1) 定位保留区 T_{orient}. 扫描 i 次找到保留区使用时间为: $Q_d n_i\% i(T_c + 2t_{\text{clk}})$. 厂商允许 NAND FLASH 存在一定数量的坏块, 一般在 2% 至 5% 之间, 同时, FLASH 设备的使用有擦写次数的限制, NAND FLASH 是 100 万次, 在空间环境中出现辐射时, 新增长使用坏块的概率 $p_{\text{ub1}} < 10^{-4}$, 连续出现两块使用坏块的概率为 $p_{\text{ub2}} = p_{\text{ub1}}^2 < 10^{-8}$ [19]. 因此, 系统寿命内连续出现两块以上使用坏块的概率极低, 假设保留区最多会出现两个坏块, 最多需扫描三次即可得到索引表位置.

$$\begin{aligned} T_{\text{orient}} &= \sum_{i=1}^{B/2} Q_d n_i\% i(T_c + 2t_{\text{clk}}) = \\ &Q_d(T_c + 2t_{\text{clk}}) \cdot (n_1\% + 2n_2\% + 3n_3\%) = \\ &Q_d(T_c + 2t_{\text{clk}}) \cdot (1 + n_2\% + 2n_3\%) < \\ &1.1 \times Q_d(T_c + 2t_{\text{clk}}). \end{aligned} \quad (2)$$

2) 读取保留区 T_{readr},

$$T_{\text{readr}} = \sum_{i=1}^{T_d} (T_c + V_i t_{\text{clk}}) \leqslant Q_d(T_c + 27Bt_{\text{clk}}). \quad (3)$$

3) 定位最后更新位置 T_{lastup},

$$T_{\text{lastup}} = N_l(T_c + t_{\text{clk}}) \leqslant Q_d(T_c + t_{\text{clk}}). \quad (4)$$

4) 恢复更新信息 T_{recover},

$$T_{\text{recover}} = N(T_c + 6t_{\text{clk}}). \quad (5)$$

因此, 本算法的启动时间 T_{startn} 为

$$T_{\text{startn}} = T_{\text{orient}} + T_{\text{readr}} + T_{\text{lastup}} + T_{\text{recover}}.$$

将式(2) ~(5) 代入式(6) 中得

$$T_{\text{startn}} = Q_d[3.1T_c + t_{\text{clk}}(3.2 + 27B)].$$

传统星载 SSR 管理算法在每页空余区写入块属性、块计数、时间码、校验码等, 每次启动时都需要扫描所有空余区, 用 T_{starto} 表示启动时间.

$$T_{\text{starto}} = Q_d BP(T_c + St_{\text{clk}}) = Q_d(BPT_c + BPSt_{\text{clk}}).$$

目前市场上常用的 NAND FLASH 中各参数的范围如下:

$$32 < P < 256,$$
$$1\,024 < B < 8\,192,$$
$$16 < S < 224,$$
$$T_c \approx 25\ \mu\text{s},$$
$$50\ \text{ps} \leqslant t_{\text{clk}} \leqslant 28.57\ \text{ps}.$$

对比式(2) 和式(3) 可得:

$$Q_d 3.1T_c \ll Q_d BPT_c,$$
$$Q_d t_{\text{clk}}(3.2 + 27B) \ll Q_d BPSt_{\text{clk}}.$$

因此, 可得 $T_{\text{startn}} \ll T_{\text{starto}}$.

5 实验结果

本文使用国产化器件搭建星载 SSR 硬件平台验证算法性能. 存储芯片使用国产珠海欧比特生产的 VDNF64G08 芯片, CPU 使用计算所研制的龙芯. FPGA 尚未国产化, 使用 Actel A3PE3000L 芯片, 工作在 64 M 晶振下. 主要存储芯片为 4 片 VDNF64G08 芯片共 256 G. 测试软件为基于 LINUX5.8 自主开发的星载 SSR 控制软件. 在该平台下使用本文提出算法与传统管理算法进行启动时间的比较试验, 并且针对相关参数进行实验、讨论. 星载相机拍摄图片通常较大, 为贴合使用需求, 测试数据选用大小为 100 K~10 M 的图像数据.

5.1 启动时间

使用测试数据在以上测试平台上分别进行功能重启和故障重启, 记录上电到索引表识别完毕的时间, 并与传统算法进行比较. 首先, 设定更新周期为 100 块, 使用测试数据将星载 SSR 存储区写满, 待到数据开始循环擦除、写入到第 550 块时, 由上位机软件发送重启指令, 得到功能重启的时间是 5.643 ms. 在同样测试条件下, 待存储区循环擦除、写入到第 550 块时, 人为掉电模拟故障, 得到的故障重启时间是 73.985 ms.

相同条件下使用传统的管理算法得到的故障重启与软件重启时间相同, 都为 50.37 s. 可见, 本文算法的启动时间显著缩短了启动时间.

5.2 更新周期

由于系统故障后启动时间与更新周期关系密切, 使用原测试数据在上述平台上进行以更新周期为变量的一组关于启动时间的测试实验. 设定更新周期分别为 10、100、500 块. 在写第 495 块时, 人为掉电模拟故障重启, 启动时间见表 3.

表3 不同更新周期下的启动时间

更新周期/块	启动时间/ms
10	6.625
100	130.349
500	753.060

故障后存留在缓存中的索引表信息丢失，缓存区中的索引表信息量与故障发生的时间以及更新周期有关. 随着更新周期增大，缓存区的信息量增多，故障发生时可能丢失信息量增多. 再次启动时需要扫描的空余区数量增多，所以系统故障启动时间变长. 虽然更新周期增加对应启动时间增长，但较长的更新周期CPU缓存内保存的索引信息量增多，索引信息发生改动时，更改索引表消耗的系统资源减少，索引表维护更便捷. 尤其时当系统内信息小范围频繁改动时，较长的更新周期更利于简化索引表从而提高启动时间. 因此，需要结合工程具体需求选择更新周期.

6 结 语

对星载SSR工作特点进行分析，提出一种可靠的快速启动算法. 该算法在原始星载SSR启动机制的基础上提出保留区概念，单独存储索引表，减少启动扫描用时; 流水更新的更新策略，提高更新效率; 双份冗余的坏块表存储方案，增加算法可靠性; 依据不同模式使用不同更新策略，充分保证其性能. 笔者在逻辑模型下对该算法进行分析并与传统算法进行比较，又在国产器件搭建的星载SSR测试平台上对其进行试验验证. 分析结果和试验结果均表明，该算法在功能性重启和故障重启条件下都能显著减少启动时间，在功能性重启条件下性能更优. 在今后的研究中，笔者将就更新周期，文件修改概率，文件大小等多种影响因素进行更多的探讨以求更优良的管理方案.

参考文献

[1] WANG J, KATZ R, SUN J, et al. SRAM based reprogrammable FPGA for space application [J]. IEEE Trans Nuclear Science, 1999, 46(6): 1728-1735.

[2] ALIA R G, BISKUP B, BRUGGER M, et al. SEU measurements and simulations in a mixed field environment [J]. IEEE Trans Nuclear Science, 2013, 60(4): 2469-2476.

[3] MCMICKELL M B, TANZILLO P, KREIDER T, et al. Rapid development of space applications with responsive digital electronics board and LabVIEW FPGA [C] // NASA/ESA Conference on Adaptive Hardware and Systems (AHS). Anaheim: [s.n.], 2010: 79-81.

[4] THOMPSON S, MYCROFT A, BRAT G, et al. Automatic in-flight repair of FPGA cosmic ray damage [C] //1st Disruption in Space Symposium. Washington: [s. n.], 2005: 287-296.

[5] KOZA J R, BENNETT F H. Genetic programming 3: darwinian invention and problem solving [M]. Berlin: Springer Netherlands, 2000: 379-381.

[6] BU K, WANG M, NIE H, et al. The optimization of the hierarchical storage system based on the hybrid SSD technology [C] //the 2nd International Conference on ISDEA. Piscataway: IEEE, 2012: 1323-1326.

[7] FABIANO M, FURANO G. NAND flash storage technology for mission-critical space applications [J]. Aerospace and Electronic Systems Magazine, IEEE, 2013, 28(9): 30-36.

[8] MARINELLA M. The future of memory [C] //2013 IEEE Aerospace Conference. Piscataway: IEEE, 2013: 1-11.

[9] WOODHOUSE D. JFFS: The journalling flash file system [EB/OL]. https://www.sourceware.org/jffs2/jffs2-html/. 2001-10-10.

[10] Aleph One Ltd, Embedded Debian. Yaffs: A NAND-Flash Filesystem [EB/OL]. http://www.aleph1.co.uk/yaffs/. [2014-5-23].

[11] LU Cao, SHI Shao. A fast mounting method for NAND flash file system 2011 [C] //3rd International Conference on Computer Research and Development (ICCRD). Shanghai: [s.n.], 2011: 416-420.

[12] YIM K S. A fast start-up technique for flash memory based computing systems [C] //Proceedings of the ACM Symposium on Applied Computing (SAC). New York: ACM, 2005: 843-849.

[13] ho CHIEN-CHUNG, HUANG PO-CHUN, KUO TEI-WEI, et al. A dram-flash index for native flash file systems [C] // International Conference on Hardware/Software Co-design and System Synthesis (CODES+ISSS). Montreal: [s.n.], 2013: 1-10.

[14] AL-SARAWI SAID F, ABBOTT DEREK, FRANZON PAUL D. A review of 3-D packaging technology [J]. IEEE Transactions on Components, Packaging and Manufacturing Technology, 1998, 21: 2-14.

[15] VDNF64G08-F 64G bit* 8 bit NAND Flash Memory Data Sheet [S]. Orbit Electronics, 2013.

[16] RIZVI S S, CHUNG Tae-Sun. An advanced and reliable initialization technique using virtual clustering for flash memory based embedded and real time systems [C] //2011 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT). Piscataway: IEEE, 2011: 509-513.

[17] RIZVI S S, CHUNG T S, AMI: an advanced endurance management technique for flash memory storage systems [J]. The International Arab Journal of Information Technology, 2011. 8(1): 1-9.

[18] 朱岩. 基于闪存的星载高速大容量存储技术的研究 [D]. 北京: 空间科学与应用研究中心, 2006.

[19] NGUYEN E N, GUERTIN S M, SWIFTG M, et al. Radiation effects on advanced flash memories [J]. IEEE Transactions on Nuclear Science, 1999, 46(6): 1744-1750.

（编辑 张 宏）